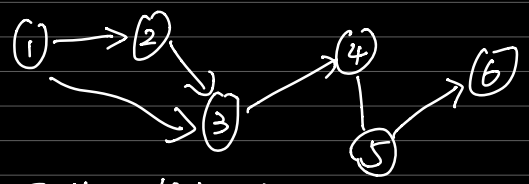
拓扑排序是针对DAG的一种判圈算法。其可以基于dfs和bfs实现.

DAG 有向无环图



其一般使用邻接数组或邻接链表存

[[1,2],[2,3],[1,3],[3,4],[4,5],[5,6]]式

1 → 2 → 3
2 → 3
3 → 4
4 → 5
5 → 6
6 → null

AOV网 数据保存在节点的DAG

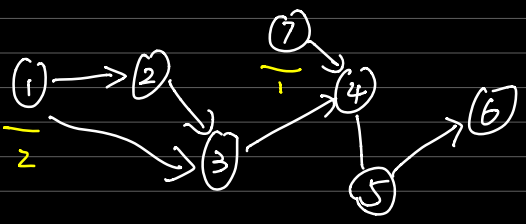
AOE网 数据保存在边的DAG

拓扑排序是为了判定有环无环，

需要注意的是拓扑排序有可能不止一种，其是一种偏序排序，而非全序排序.

(偏序指的是部分有序而非全部有序，全序指的是全有序有唯一的序列)

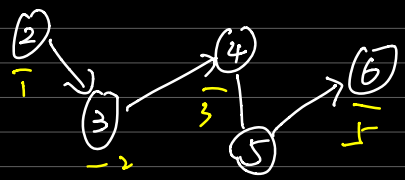
基于bfs的拓扑排序(正常思路)



bfs

寻找入度为0的节点，删除节点(更新其它点的入度)，直至无节点为止

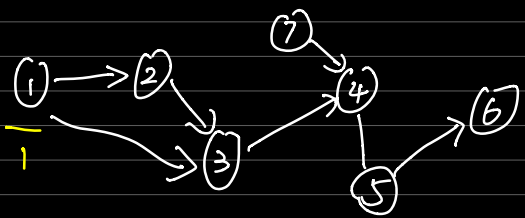
7 → 1 → 2 → 3 → 4 → 5 → 6



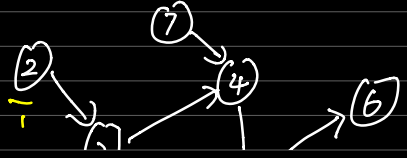
一种排序法

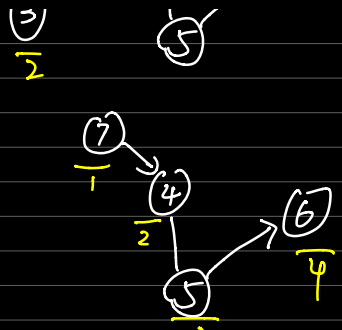
此思路的搜索顺序和bfs一致。

其和bfs的不同在于多点开始



7 → 1 → 2 → 3 → 4 → 5 → 6





另外一种可能的排序

其中多个同时入度为0的节点的写法会对很多

```
while (!queue.isEmpty()){
    int node = queue.poll();
    for (int ncd : data.getNode(node)){ // 与节点相连其它节点
        degrees[ncd]--;
        if(degrees[ncd] == 0) queue.offer(ncd);

    }
}
```

只要理解了其原理，dfs也是可以进行拓扑排序的，其实现思路在于3种状态的设计。

未搜索：即初始状态。

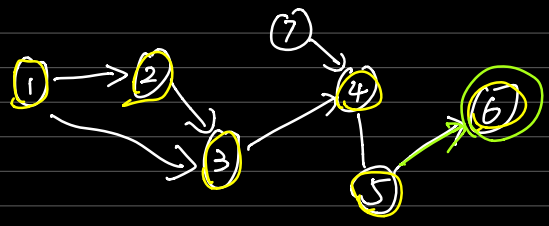
搜索中：如果找到了搜索中的节点，那么则有环，否则往下找的标记为此

已完成：已经找完的节点

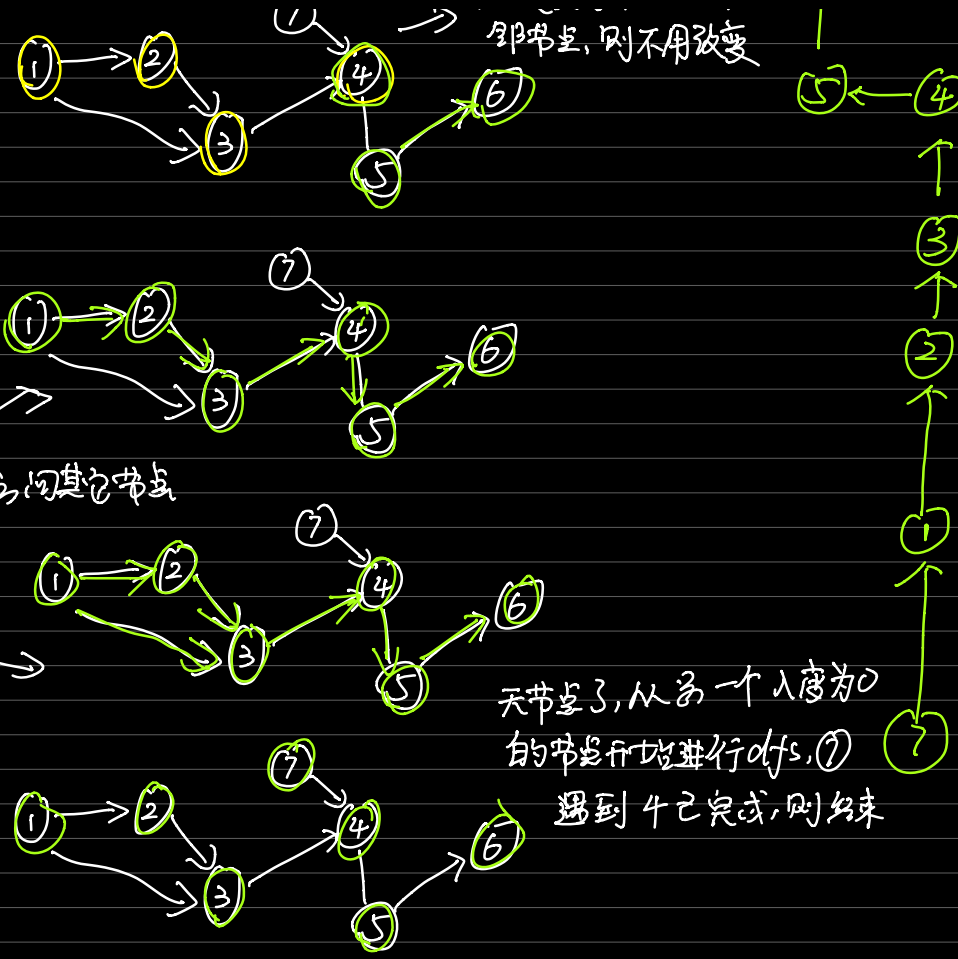
dfs：

无其它相邻节点，终来返回

访问其它节点



此外因为4无其它相


邻接点，则不用改变
访问其它节点
无节点了，从另一个入度为0
的节点开始进行dfs，①
遇到 4 已完成，则结束